PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-228387
(43)Date of publication of application : 25.08.1998

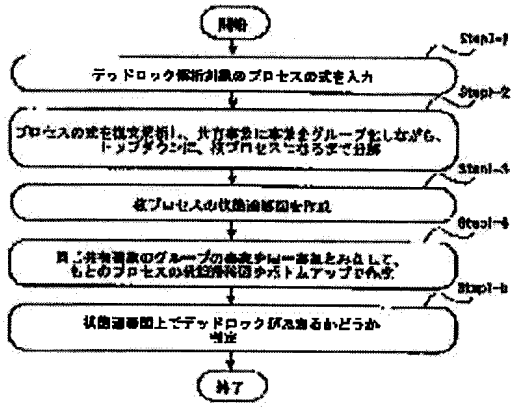
(51)Int.Cl. G06F 9/46
G06F 11/30
G06F 15/16

(21)Application number : 09-030061
(22)Date of filing : 14.02.1997

(71)Applicant : HITACHI LTD
(72)Inventor : KASAHARA TAKAYASU
NAKANO TOSHIHIKO

(54) SPECIFICATION VERIFICATION SYSTEM

(57)Abstract:
PROBLEM TO BE SOLVED: To reduce calculation volume required for judging whether a program synthesized from program components generates dead lock or not even when the program is used for parallel processing.
SOLUTION: In Step 1-1, the expression of a process to be analyzed at its dead lock is inputted, and in Step 1-2, syntax analysis for the process expression is executed, the process is decomposed up to kernel processes while grouping the events as a shared event to prepare a block diagram of processes. In Step 1-3, 1-4, the state transition diagram of a synthetic process is prepared from the process block diagram in a bottom up state, and in Step 1-5, dead lock is judged based on the generated state transition diagram of the synthetic process.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－228387

(43)公開日　平成10年(1998)８月25日

(51)Int.Cl.[6]		識別記号	FI		
Ｇ０６Ｆ	9/46	３４０	Ｇ０６Ｆ	9/46	３４０Ｇ
	11/30	３０５		11/30	３０５Ｇ
	15/16	４７０		15/16	４７０Ａ

審査請求　未請求　請求項の数４　ＯＬ　（全 10 頁）

(21)出願番号　特願平9－30061

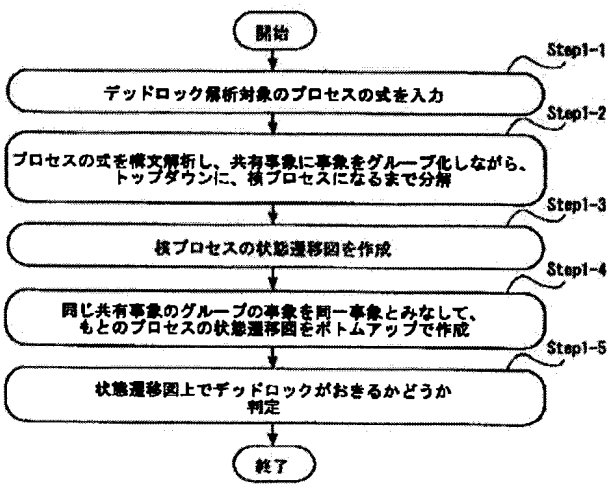
(22)出願日　平成９年(1997)２月14日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地
(72)発明者　笠原　孝保
茨城県日立市大みか町七丁目２番１号　株式会社日立製作所電力・電機開発本部内
(72)発明者　中野　利彦
茨城県日立市大みか町五丁目２番１号　株式会社日立製作所大みか工場内
(74)代理人　弁理士　小川　勝男

(54)【発明の名称】　仕様検証方式

(57)【要約】
【課題】部品化されたプログラムが合成されて、並列処理に用いられてもデッドロックをおこさないかどうかの判定に要する計算量を削減する。
【解決手段】Step１－１ではデットロックの解析対象のプロセスの式を入力しStep１－２でプロセスの式を構文解析し共有事象に事象をグループ化しながら核プロセスになるまで分解し、プロセスの構造図を作成する。Step１－３，４でプロセス構造図からボトムアップに合成プロセスの状態遷移図を作成し、Step１－５で生成した合成プロセスの状態遷移図をもとにデッドロックの判定を行う。

図　1



1

【特許請求の範囲】

【請求項１】複数システムが並行動作する場合に発生しうる、システムが相互に信号待ちの状態になって動作しなくなるデッドロック現象の発生の可能性の有無を検証する仕様検証システムにおいて、上記複数システムが動作するプロセスで発生する事象を上記複数システムが共有する共通処理・動作である共有事象に基づいて、システムの動作系列で直前に発生する共有事象にまとめてグループ化し、グループ化された事象をそのグループを代表する共有事象によって名付けられた一つの事象とみなしてデッドロックの有無の判定を行うことを特徴とする仕様検証方式。

【請求項２】複数システムが並行動作する場合に発生しうる、システムが相互に信号待ちの状態になって動作しなくなるデッドロック現象の発生の可能性の有無を検証する仕様検証システムにおいて、上記複数システムが動作するプロセスで発生する事象を上記複数システムが共有する共通処理・動作である共有事象に基づいて、システムの動作系列で直前に発生する共有事象にまとめてグループ化し、グループ化された事象をそのグループを代表する共有事象によって名付けられた一つの事象とみなして、検証結果を説明するための状態遷移図として、状態遷移図における遷移を、事象のグループを代表する共有事象によりラベル付けしたものを用いることを特徴とする仕様検証方式。

【請求項３】あらかじめ用意された仕様部品を組み合わせて、複数のシステムが並行動作するシステムを構築する設計支援システムにおいて、用意する部品の挙動を入出力変数と内部変数の区別なく記述することができ、デッドロックの検証を行うことを特徴とする仕様検証方式。

【請求項４】あらかじめ用意された仕様部品を組み合わせて、複数のシステムが並行動作するシステムを構築する設計支援システムにおいて、用意する部品の挙動を入出力変数と内部変数の区別なく記述し、デッドロックの検証を行うことを特徴とする仕様検証方式。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は仕様検証方式に関する。

【０００２】

【従来の技術】従来技術の解説は、添付した解説記事に＊

2

＊まとめられている（情報処理学会誌Vol．３５　No．８，pp７３６－７４１）。プロセス代数を用いた仕様は、本解説記事の（１）式のように並列動作を論理的に厳密に定義するためのプロセス代数の仕様記述言語（この例ではプロセス代数の一つであるＣＣＳが用いられている。）で記述され、これを図２のような遷移グラフに展開して、デッドロックのチェック等の検証を行う。これを、図２の原子力や火力等のプラントの運転支援・監視を行うシステムを仕様部品を用いて構築する際の、デッドロックの有無の判定を例にとって説明する（ここでは、仕様をプログラムも含めた広い意味で考える）。ここで、デッドロックとは、二つのサブシステムが相互に信号待ちの状態になって、動作しなくなることを指す。

【０００３】構築するプラント運転支援・監視システムは、図２に示すように、プラントからの制御信号を受け取り、センサ信号を処理してマンマシンインターフェースに操作指示やセンサ値などの情報を含んだ表示データを送るものである。最終的には運転員はマンマシンインターフェースからセンサ信号や運転操作の指示を受け取ることになる。ここでは、このマンマシンインターフェースの部分は除外した部分をプラント運転支援・監視システムと呼ぶことにする。以下の三つのプログラム部品を組み合わせてプラント運転支援・監視システムを構築する場合を考える。（１）センサからの信号と、他のプログラム部品から受け取った操作指示のデータを加工して表示データとし、マンマシンインターフェースに送るとともに、前処理したセンサ信号を他のプログラム部品に送る運転支援・監視情報制御部のプログラム部品。（２）前処理されたセンサ信号を用いて運転状況を判定する運転状況判定部品、（３）運転状況判定結果を受け取って対応する操作指示を出す対応操作推論部品。これら三つのプログラム部品の仕様は列処理システムを記述するのにもっとも一般的な記述方法の一つであるＣＳＰの記法により記述されている。それらを数１，数２，数３に示す。これらは各々上記(１)，(２)，(３)の部品に対応する。また、数４は、プログラム部品（２），（３）を組み合わせて作成した状況判定・対応操作決定部に対応する仕様であり、数５は仕様部品(１)，(２)，(３)を組み合わせてできるプラント運転支援・監視システムの仕様である。

【０００４】

$$P = a \rightarrow d \rightarrow b \rightarrow d \rightarrow a \rightarrow a \rightarrow a \rightarrow P \quad \cdots (数1)$$

$$Q = a \rightarrow c \rightarrow Q \quad \cdots (数2)$$

$$R = c \rightarrow b \rightarrow R \quad \cdots (数3)$$

$$S = Q \ [\ |c|\ ]\ R \quad \cdots (数4)$$

$$T = P \ [\ |a, b|\ ]\ S \quad \cdots (数5)$$

また、数１から数５までに用いられている記号と図２中に記述されている“操作指示”，“運転状況判定結果”などのデータの授受を示す事象との対応関係を図３に示す。例えば、数１のプロセスＰは、運転支援・監視情報制御部のプログラム部品が実行するプロセスを示す。ただし、ここでは、簡単のため、各プロセス同士のデータ

・信号の授受は同時に起き共有事象をなすとしている。プロセスPは、センサから受け取った信号を加工して前処理済みのセンサ信号を発生し（事象a）、表示データを発生し（事象d）、操作指示を受け取り（事象b）、再びその内容を加工して、表示データを発生し（事象d）、次にセンサから受け取った信号を3回続けて加工して発生し（事象a）、これを繰り返すプログラム部品である。同様に、数2のプログラム部品は運転状況判定部のプログラム部品で、前処理されたセンサ信号を受け取り（事象a）、これを用いて運転状況判定し、運転状況判定結果を発信する（事象c）。以上の処理を繰り返すプログラム部品である。また、数3のプログラム部品は、対応操作推論部に相当し、運転状況判定結果を受け取り（事象c）、これに基づいて推論を実行して操作指示を発信する（事象b）。これを繰り返すプロセスである。これらの仕様は事象の発生する順序を規定したものであり、データの内容には触れていない。以上三つのプロセスはこれ以上分解できないプロセスで核プロセスと呼ばれる。これに対して、合成プロセスとはこれらの核プロセスを並列に動かすプロセスである。例えば、数4のプロセスは、核プロセスQと核プロセスRとを事象cを共有事象として並列動作する合成プロセスである。ここで共有事象とは、二つのプロセスで同時に発生しなければならないプロセスである。今の例では、はじめに、プロセスQで事象“a”が発生し、次に事象“c”が発生したらプロセスRでも同時に事象“c”が発生すると規定したものである。この合成プロセスSは図2の状況判定・対応操作部の動作を表わすプロセスであり、前処理済みセンサ信号を受け取ったら、運転状況を判定し、それに基づいて操作指示を決めて出力するサブシステムである。一方、数5は数1から数3までのプログラム部品を合成して図2の運転支援・監視システム全体を実現することを意図した合成プロセスTで、（a）前処理済みのセンサ信号を発生して（事象“a”）、そのデータを加工して(2)表示信号データとして発信し（事象“d”）、一方、その信号から運転状況を判定して対応操作を決め（事象“b”）、対応操作を表示用に加工して出力し（事象“d”）、そのあと、前処理済み信号を3回発生する（事象“a”）。以上の処理を繰り返すプロセスである。

【0005】このように部品の組み合わせだけで作成した制御プログラムが、システム全体を思いどうりに制御できるかどうかをチェックするには、従来、個々の部品の状態遷移図をボトムアップに合成して全体の状態遷移図を作成することで行われてきた。例えば、デッドロックの発生がないことは、システム設計の重要な用件である。従来のデッドロック解析の方法を説明するために、数5で表わされるプロセスTがデッドロックするかどうかを判定する従来方法を説明する。図4は、デッドロックを判定するための概略処理手順を示している。図中のStep3－1では、通常の字句解析技術により、数1から数5までの式を入力し、デッドロックの解析対象のプロセスがプロセスT（数5）であることを指定する。次に入力された式を構文解析し、核プロセスにトップダウンに分解する。トップダウンに分解するとは、プロレスが、すでに説明した並列結合の演算子［ |…| ］などで二つのプロセスから合成されたものであれば、個々のプロセスに分解することを指す。また、核プロセスとは、このような方法で、これ以上分解できないプロセスを指す。このような分解も通常の構文解析技術により可能である。分解した結果できるプロセスの構造図を図5に示す。プロセスの構造図を木構造をしており、木の葉にあたるものが核プロセスで、この場合、プロセスP，Q，Rがそれに当たる。図中の木のノードに記述したのはサブツリーから合成された合成プロセスである。このような木構造のデータ構造はC，Lisp など一般の高級言語で実現可能であるので以後の説明は、木構造のレベルでの記述にとどめる。次にStep3－3で核プロセスの状態遷移図を作成する。図6は核プロセスP，Q，R各々の状態遷移図を示した。状態遷移図の状態を区別するために適当な名称（P1，P2，…など）をつけている。出発点の状態のない矢印の行き先になっている状態は初期状態である。プロセスP，Q，Rの初期状態はそれぞれ、P0，Q0，R0である。また、矢印の上に書かれた文字列はその文字列の事象を発生して、状態が遷移することを示す。たとえば、核プロセスPでは、初期状態P0の次に、事象aを発生して状態P1となる。このような状態遷移図と等価なデータ構造，オペレーションも一般の高級言語で容易に実現できる。次に、Step3－4で各、核プロセスの状態遷移図をプロセスの合成図に基づいてボトムアップに順次作成する。図7に合成プロセスの状態遷移図を作成する手順を示す。これを、図5のなかで、核プロセスQとRから、共通事象をcとする合成プロセスQ［ |c| ］Rを作成する例によって説明する。合成結果の状態遷移図は図8である。まず、Step6－1では合成もとのプロセスの初期状態の組を合成プロセスでの初期状態とする。これが、図8の左端の状態（Q0，R0）に相当する。この状態がStep6－2で状態の待ち行列の先頭におかれ、次のStep6－3で取り出され、（Qt，Rt）＝（Q0，R0）と置かれる。次にStep6－4で、Q0からの共通事象c以外の遷移はaによるものがあり、この遷移（a，ε）によって、新たな状態（Q1，R0）が生成される。ここで、εは何も事象が発生しないことを示し、R0にとっては、aが発生しても関係なく、もとの状態R0のままであることを示す。遷移を起こす事象はこのように、事象の組になっているが、これも事象aによる遷移と見なす。プロセスの構造によっては、事象の組のいれ子はいくらでも深くなる。Q0からの共通事象c以外の遷移はこれだけであり、この状態が待ち行列に追加される。一方、プロセス

Ｒ０からの共有事象ｃ以外の遷移はない。次にStep６－５で、共有事象ｃによる遷移がＱ０，Ｒ０で共通におきるかどうか調べると、Ｒ０からの遷移はあるが、Ｑ０からの遷移はない。したがって、Step６－６で新たに追加される状態はない。また、すでに(ａ，ε)による遷移が見つかっているので状態（Ｑ０，Ｒ０）はデッドロック状態ではない。次にStep６－７で待ち行列を見ると、先頭に（Ｑ１，Ｒ０）が入っている。そこで、Step６－４で、この（Ｑ１，Ｒ０）を待ち行列から取り出しStep６－４を実行すると、共有事象以外の遷移は状態Ｑ１，Ｒ０ともないので、遷移は発生しない。次にStep６－５を実行すると、共有事象ｃにより、Ｑ１は状態Ｑ０に、Ｒ０の状態はＲ１に遷移するので、事象（ｃ，ｃ）による新たな状態（Ｑ０，Ｒ１）への遷移を状態遷移図に書き込み、状態の待ち行列に（Ｑ０，Ｒ１）を追加する。次にStep６－６を実行しても、すでに（Ｑ０，Ｒ１）からの遷移は見つかっているので（Ｑ０，Ｒ１）はデッドロック状態ではない。次にStep６－７を実行すると、状態の待ち行列に、新たにできた状態（Ｑ０，Ｒ１）がある。そこで、Step６－３で、状態（Ｑ０，Ｒ１）を取り出す。次にStep６－４で、Ｑ０からの共有事象以外による遷移をさがすと、ａによる遷移ができる。そこで、この事象（ａ，ε）により生成する状態（Ｑ１，Ｒ１）はまだ状態遷移図中にないので、状態の待ち行列に追加する。一方、Ｒ１からの共有事象以外による状態の遷移として、ｂによる状態遷系がある。この事象（ε，ｂ）による状態遷移により生成する状態（Ｑ０，Ｒ０）は、すでに状態遷移図中に存在するので矢印を状態遷系図中に書き込むだけでよい。Step６－５では共有事象による遷移は見つからない。すでに遷移先が見つかっているのでデッドロックは生じない。

【０００６】この段階でStep６－７の判定を行うと、待ち行列には新たにできた状態（Ｑ１，Ｒ１）があるので空でない。そこで、Step６－３で状態（Ｑ１，Ｒ１）が取り出される。次にStep６－４でＱ１からの共有事象以外による遷移はないが、Ｒ１からの非共有事象ｂによる遷移がある。ただし、（ε，ｂ）によって生成する状態（Ｑ１，Ｒ０）はすでに状態遷移図中にあるので、矢印をつけるだけでよい。次にStep６－５で共有事象による遷移はない。Step６－６で状態（Ｑ１，Ｒ１）からの遷移はすでに見つかっているのでデッドロック状態ではない。次にStep６－７で、状態の待ち行列は空なので合成プロセスの生成は終了する。

【０００７】同様にして、プロセスＰとプロセスＱ［｜ｃ｜］Ｒのａ，ｂを共有事象とする合成プロセスの状態遷移図を作成した結果が図９である。これで、図４のStep３－４の対象となるプロセスの状態遷移図が完成した。状態遷移図の中で、＊印の着いた状態（Ｐ６，（Ｑ１，Ｒ１））は、次の状態に遷移することのできないデッドロック状態である。次にStep３－５で、状態遷移図中にデッドロック状態があるかどうかを判断する。この場合は、状態遷移図中にデッドロック状態があるのでデッドロックありと判定される。

【０００８】

【発明が解決しようとする課題】従来技術では、生じる事象の数が多くなるにつれ、最終的にできる状態遷移図中の状態数が指数関数的に増大し、その結果、デッドロックの判定のために状態遷移図を作成する際に要する計算量も指数関数的に増大する。このため、大規模な問題でデッドロックの判定を行うのは困難であった。また、検証結果を人間が確認する際に、状態遷移図を見ても状態数が多すぎて確認することが困難であるという問題があった。このような問題があるため、従来技術では仕様部品を、あらかじめ入出力変数を指定し、他の変数は内部変数として仕様に現われないようにし、部品の動作自体は、入出力変数のタイミングや値の関係として、より抽象的な記述をする必要があった。これにより、部品を組み合わせる時の検証における計算量を減らしていた。例えば、数１で、入出力変数をａ，ｂだけと限定すると、抽象化された仕様部品は、内部変数ｄを無視したプロセスＰ′＝ａ－＞ｂ－＞ａ－＞ａ－＞ａ－＞Ｐ′と記述され、これを用いると検証はより簡単なものになる。しかし、この従来手法では、仕様部品が利用できる範囲が限定的なものになる。

【０００９】本発明の目的は、仕様部品の動作を入出力変数と内部変数の区別なく記述することにより、仕様部品の品揃えを容易化するとともに、デッドロックに要する計算量を減らし、検証結果を人間が確認することを容易にすることにある。

【００１０】

【課題を解決するための手段】上記目的は、プロセスの構造図を作成する際に非共有事象を直前に発生する共有事象によってグループ化する処理手段、およびグループ化された事象を一つの共有事象とみなして状態遷移図の作成およびデッドロックのチェックを行う処理手段により達成される。

【００１１】

【発明の実施の形態】以下、図１を用いて本発明の一実施例を説明する。プロセスＰ［｜ａ，ｂ｜］（Ｑ［｜ｃ｜］Ｒ）のデッドロックを判定する場合を例にとって説明する。図１はプロセスのデットロックを本発明の手順によって判定する手順を示しており、従来技術による方法を示した図４と対応するものである。本手法による検証を前提としているため、用意する仕様部品は数１～数３にあるように入出力変数と内部変数をあらかじめ区別なくプロセス代数ＣＳＰにより記述しておくことができる。Step１－１では、デッドロックの解析の対象となる式（今の場合Ｐ［｜ａ，ｂ｜］（Ｑ［｜ｃ｜］Ｒ））が入力される。

【００１２】次にStep１－２で入力したプロセスの式を

構文解析し、共有事象に事象をグループ化しながら各プロセスに分解する。図１０はこの構文解析の結果生成した構文木を示す。この手順は以下のようになる。まず、共有事象を格納するリストＬを空リストに初期化する。次に、もとのプロセスをプロセスＰとプロセス（Ｑ［｜ｃ｜］Ｒ）に分解する。その際、共有事象“ａ”と“ｂ”を共有事象のリストＬに追加する。その結果、Ｌ＝［“ａ”，“ｂ”］となる。次に分解された各プロセスが、核プロセスかどうか判定する。プロセスＰは、核プロセスであるので、共有事象によりグループ化したプロセスに書き直す。その結果、プロセスＰはＰ＝ａ－＞ｂ－＞ａ－＞ｄ－＞Ｐから、Ｐ＝ａ(Ｐ１)－＞ｂ(Ｐ１)－＞ａ(Ｐ２)－＞Ｐに書き変えられる。ここで、ａ(Ｐ１)共有事象ａのプロセスＰにおける１番めの事象グループであることを示す。ａ(Ｐ１)には事象のリスト［ａ］を対応づける。同様にｂ(Ｐ１)は共有事象ｂのプロセスＰにおける２番めの事象グループであることを示し、事象のリスト［ｂ］を対応づける。ａ(Ｐ２)は共有事象ａのプロセスＰにおける２番めの事象グループであることを示し、ａ(Ｐ２)には事象のリスト［ａ，ｄ］を対応づける。一方、プロセスＱ［｜ｃ｜］Ｒは核プロセスでないので、さらに分解を続ける。まず、この合成プロセスの共有事象“ｃ”を共有事象のリストＬに追加しその結果Ｌ＝［“ｃ”，“ａ”，“ｂ”］となる。分解されたプロセスＱとＲは核プロセスなので、Ｐの場合と同様の方法で書き直される。その結果、Ｑ＝ａ(Ｑ１)－＞ｃ(Ｑ１)－＞Ｑ，Ｒ＝ｃ(Ｒ１)－＞ｂ(Ｒ２)－＞Ｒとなる。次に、Ｓｔｅｐ１－３で、プロセスの構文木中の核プロセスの状態遷移図を作る。この方法は従来例と同じで、図１１のようになる。次に図１のＳｔｅｐ１－４でもとのプロセスの状態遷移図を図１０のプロセス構造図の核プロセスの状態遷移図からボトムアップに合成していくことにより作成する。その際、同じ共有事象のグループに属する事象は、同じ事象と見なす。最初にプロセスＱの核プロセスとプロセスＲの核プロセスから合成プロセスＱ［｜ｃ｜］Ｒを作成する。この合成方法も従来方法と同じであるが、ただし、変数ｃ(Ｒ１)と変数ｃ(Ｑ１)を同一の共有事象と見なすところが異なる。その結果、状態遷移図は図１２のようになる。同様にして、プロセスＰの状態遷移図とプロセスＱ［｜ｃ｜］Ｒの状態遷移図からプロセスＰ［｜ａ，ｂ｜］の状態遷移図を作成すると図１３のようになり、状態（Ｐ４′，(Ｑ１′，Ｒ１′))に到達するとデッドロックが発生することがわかる。この状態はもとの核プロセスでの対応を考えれば、従来手法による合成プロセスのデッドロック状態（Ｐ６，(Ｑ１，Ｒ１)）に対応する。また、変数の書き換えのオーバーヘッドを無視すれば、デッドロックの判定処理に要する計算量は、状態遷移図の状態と遷移を示す矢印の数にほぼ比例すると考えられる。従来手法の状態遷移図では、状態数９，矢印の数は、初期状態への矢印も含めると、９となる。一方、本発明による状態遷移図１３では、状態数７，矢印の数も７で計算量が減っていることがわかる。また、デッドロックの発生をユーザに説明するための図は、従来手法では図９，本発明では図１３のようになる。

【００１３】これらの説明は、図１４に示した装置構成によって、検証を行うためのＣＰＵとメモリの乗った演算処理装置と演算処理装置上にプログラムされた表示のための表示プログラムにより、演算処理装置に接続した表示装置により表示することができ、状態遷移図の簡単な本発明の表示の方が理解が容易である。

【００１４】

【発明の効果】本発明によれば、デッドロックの検証に要する計算量を減らし、また、検証結果をユーザに分かりやすく表示することができる。また、ユーザは品揃えする部品を入出力変数と内部変数の区別なく記述した形で用意することができる。

【図面の簡単な説明】

【図１】本発明によるデッドロックのフローチャート。

【図２】運転支援・監視システムのブロック図。

【図３】プロセスの名称と処理内容との対応関係の説明図。

【図４】デッドロックを判定するための概略処理のフローチャート。

【図５】プロセスの構造の説明図。

【図６】核プロセスＰ，Ｑ，Ｒの状態遷移の説明図。

【図７】合成プロセスの状態遷移図作成のフローチャート。

【図８】合成プロセスＱ［｜ｃ｜］Ｒの状態遷移の説明図。

【図９】合成プロセスＰ［｜ａ，ｂ｜］(Ｑ［｜ｃ｜］Ｒ）の状態遷移の説明図。

【図１０】本発明によるプロセス構造の説明図。

【図１１】本発明による核プロセスＰ，Ｑ，Ｒの状態遷移の説明図。

【図１２】本発明による合成プロセスＱ［｜ｃ｜］Ｒの状態遷移の説明図。

【図１３】本発明の一実施例方法を示すタイミングチャート。

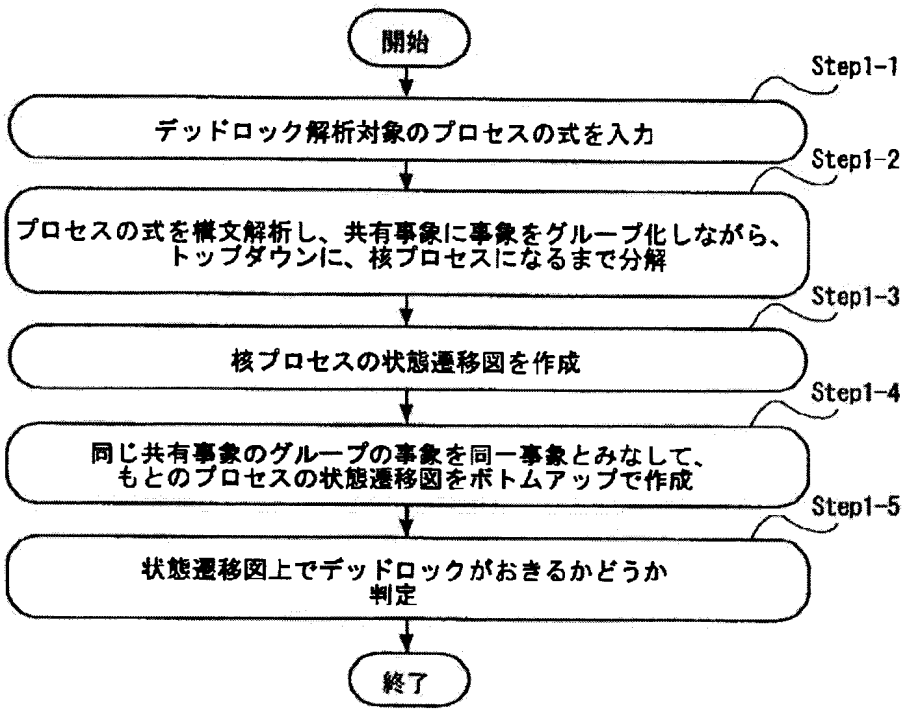
【図１４】運転・支援システムの装置の説明図。

【符号の説明】

Ｓｔｅｐ１－１…デッドロック解析対象プロセスの入力、Ｓｔｅｐ１－３…核プロセスの状態遷移図作成。

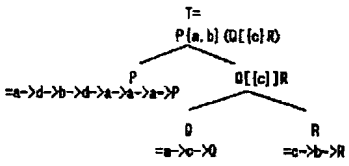
【図１】

図　1



【図５】

図　5



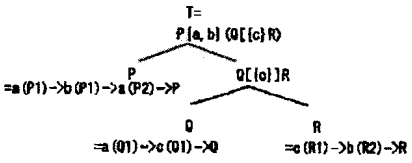
【図３】

図　3

プロセス名称	説明	事象の説明
P	運転支援・監視情報制御部の動き	a:前処理済みセンサー信号の授受 b:操作指示の授受 d:表示データの授受
Q	運転状況判定部の動き	a:前処理済みセンサー信号の授受 c:運転状況判定結果の授受
R	対応操作推論部の動き	b:操作指示の授受 c:運転状況判定結果の授受

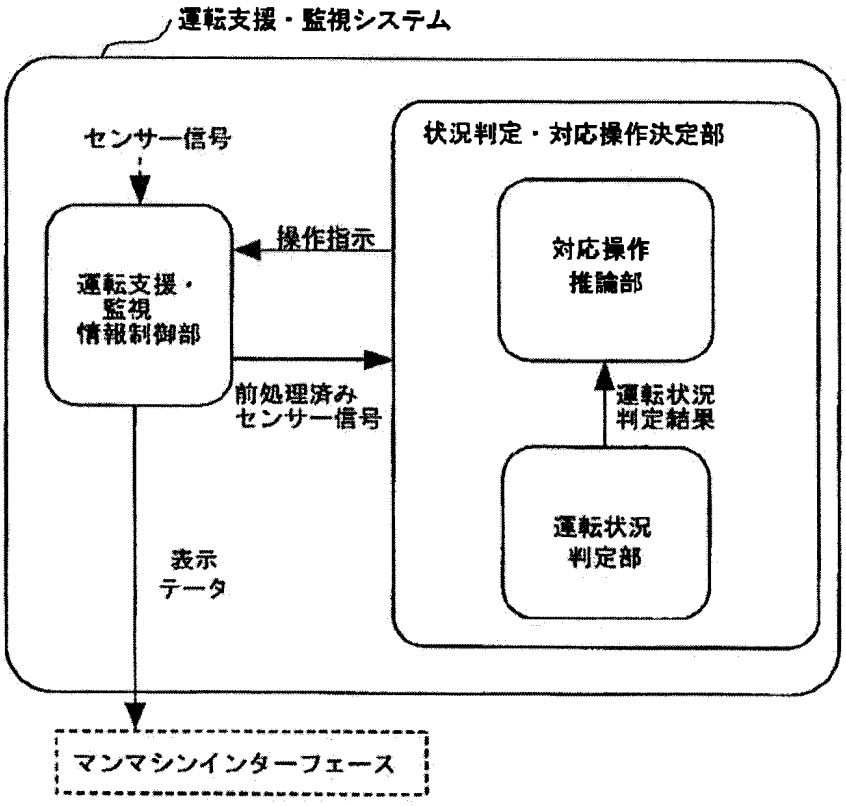
【図１０】

図　10



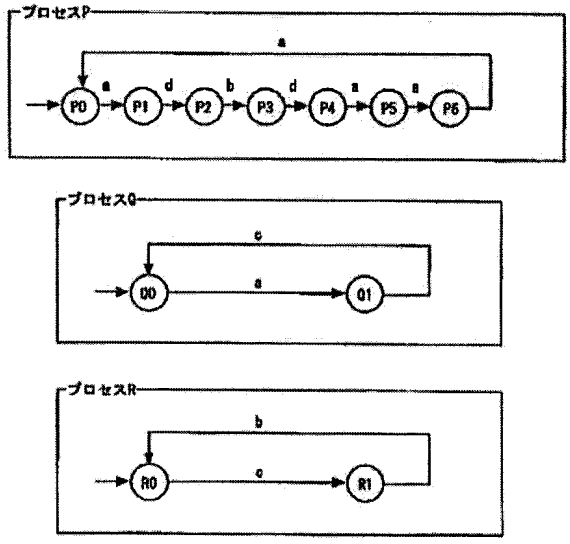
【図２】

図　２



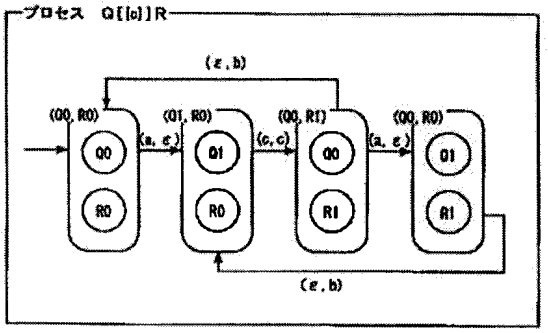
【図６】

図　６



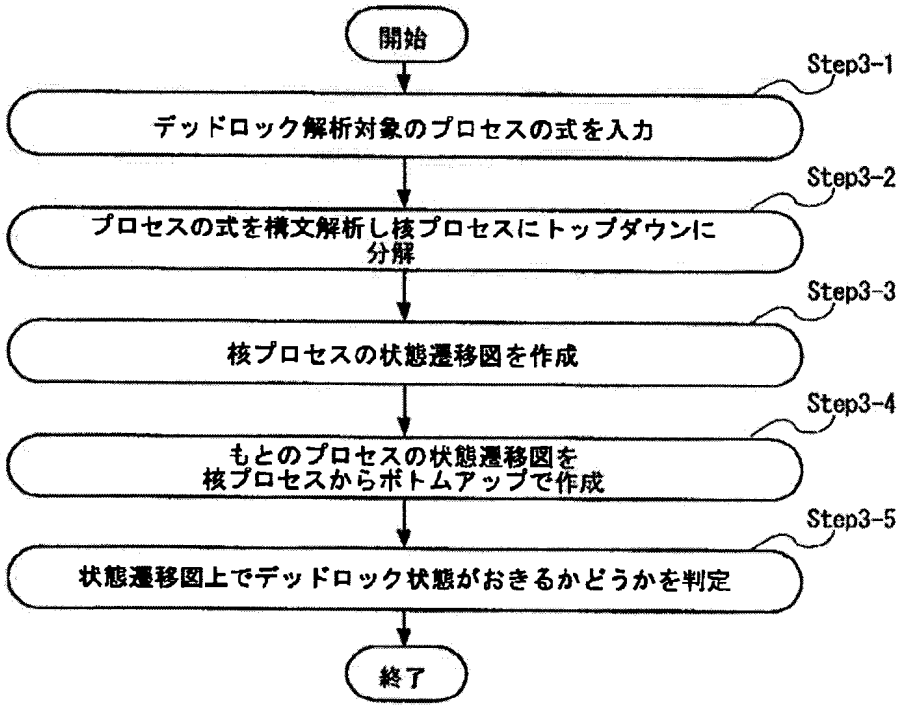
【図８】

図　８



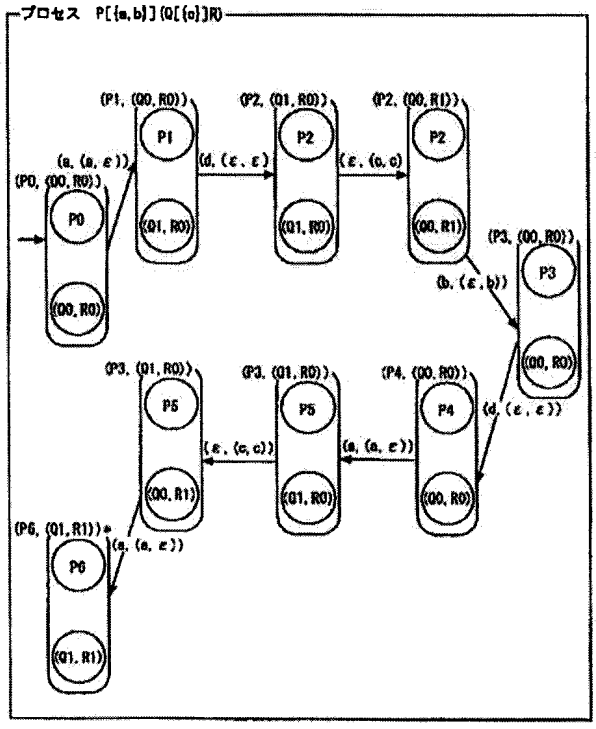
【図４】

図　4



【図９】

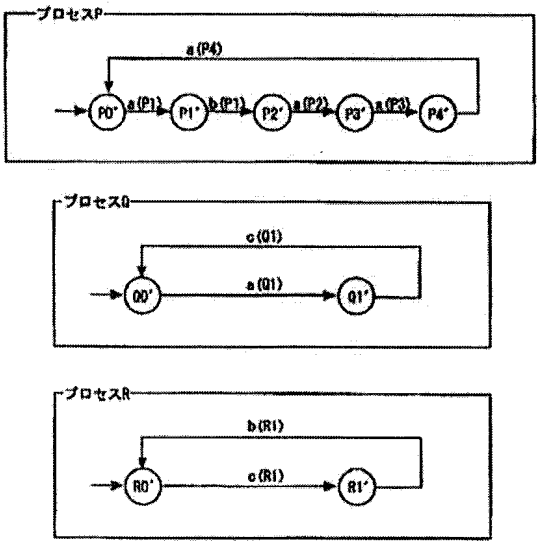
図　9



※はデッドロック状態を表す

【図１１】

図　11

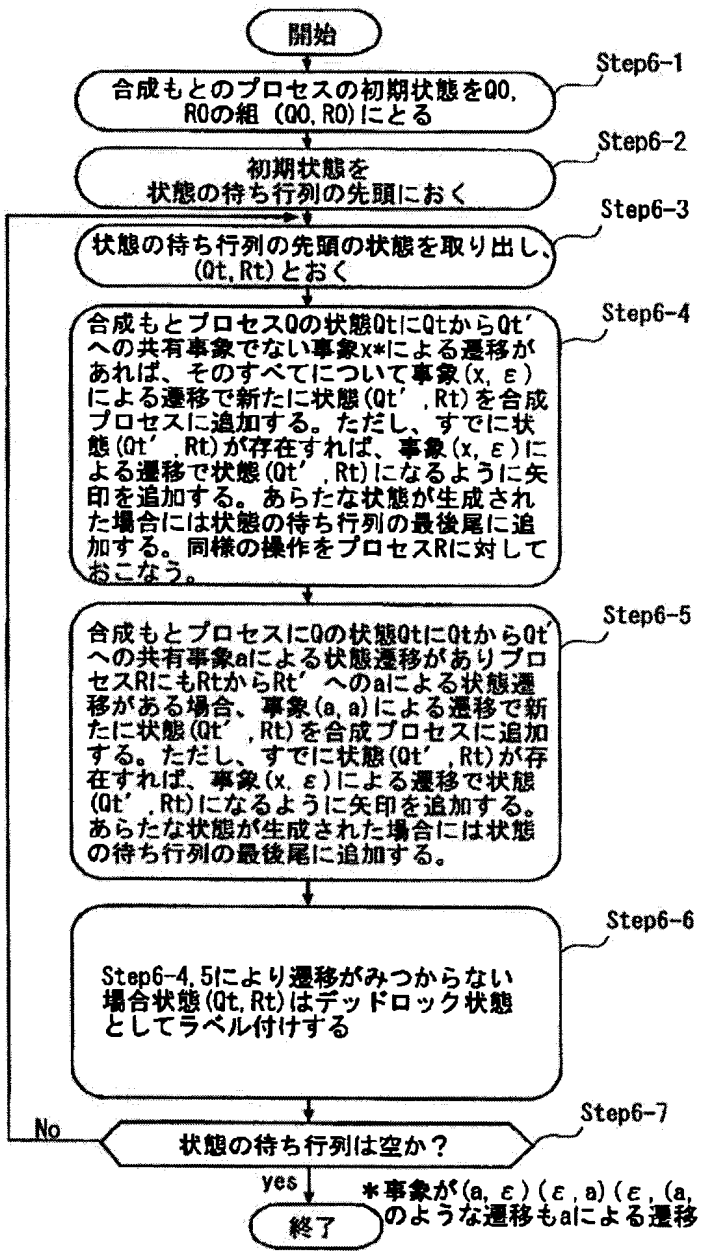


ただし、

a(P1)=[a, d] ; b(P1)=[b, d] ; a(P2)=[a] ; a(P3)=[a] ; a(P4)=[a] ;
a(Q1)=[a] ; c(Q1)=[c] ;
c(R1)=[c] ; b(R1)=[b]

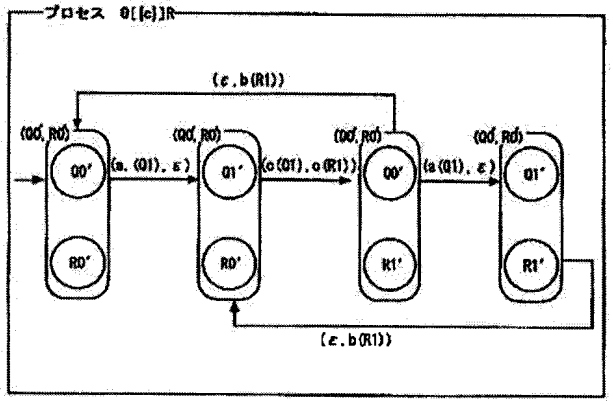
【図７】

図　7


開始
Step6-1
合成もとのプロセスの初期状態をQ0, R0の組 (Q0, R0)にとる
Step6-2
初期状態を状態の待ち行列の先頭におく
Step6-3
状態の待ち行列の先頭の状態を取り出し、(Qt, Rt)とおく
Step6-4
合成もとプロセスQの状態QtにQtからQt'への共有事象でない事象x*による遷移があれば、そのすべてについて事象(x, ε)による遷移で新たに状態(Qt', Rt)を合成プロセスに追加する。ただし、すでに状態(Qt', Rt)が存在すれば、事象(x, ε)による遷移で状態(Qt', Rt)になるように矢印を追加する。あらたな状態が生成された場合には状態の待ち行列の最後尾に追加する。同様の操作をプロセスRに対しておこなう。
Step6-5
合成もとプロセスにQの状態QtにQtからQt'への共有事象aによる状態遷移がありプロセスRにもRtからRt'へのaによる状態遷移がある場合、事象(a, a)による遷移で新たに状態(Qt', Rt)を合成プロセスに追加する。ただし、すでに状態(Qt', Rt)が存在すれば、事象(x, ε)による遷移で状態(Qt', Rt)になるように矢印を追加する。あらたな状態が生成された場合には状態の待ち行列の最後尾に追加する。
Step6-6
Step6-4, 5により遷移がみつからない場合状態(Qt, Rt)はデッドロック状態としてラベル付けする
Step6-7
No
状態の待ち行列は空か？
yes
終了
＊事象が(a, ε)(ε, a)(ε, (a, ε))のような遷移もaによる遷移とみなす。

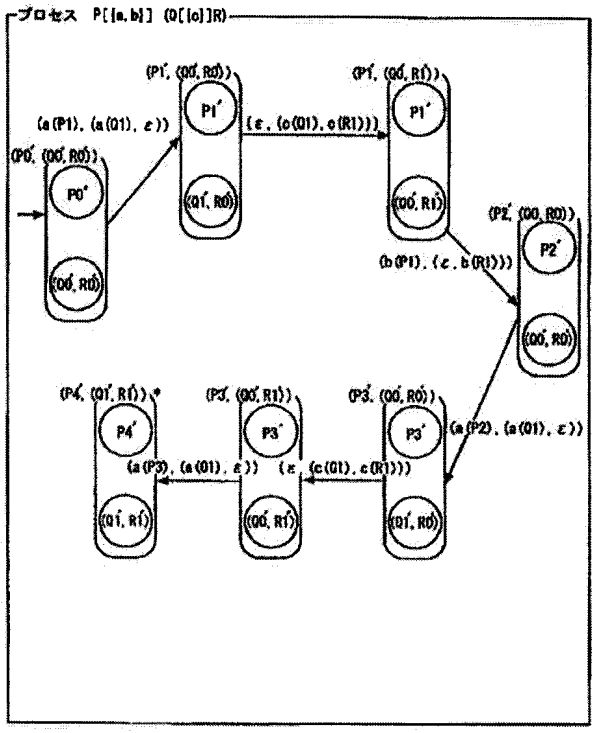
【図１２】

図　12



【図１３】

図　13



＊はデッドロック状態をあらわす

【図１４】

図　14